DOCUMENT RESUME

ED 428 565 FL 025 764

AUTHOR Sypniewski, Bernard Paul
TITLE Linguistic Types and the Valence of Operators in Applicative Universal Grammar.
PUB DATE 1998-00-00
NOTE 9p.
PUB TYPE Reports - Descriptive (141)
EDRS PRICE MF01/PC01 Plus Postage.
DESCRIPTORS *Grammar; *Language Patterns; Language Research; *Language Universals; *Linguistic Theory
IDENTIFIERS Applicative Universal Grammar

ABSTRACT

The relationship between linguistic types and the valence of operators on the genotype level of Applicative Universal Grammar (AUG) is examined. Assuming that the "t" and "s" types may be treated as zero-place operators, a relationship is found between the valence of an operator and its genotype, which explains the difference between types operators, e.g., modifiers and predicates. The motivation for this work is the construction of a parser that correctly identifies superposition, a theory of AUG that posits that, in certain circumstances, an expression may have a stratified type rather than a simple type. (MSE)

**********
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
**********

ED 428 565

# Linguistic Types and the Valence of Operators in Applicative Universal Grammar

by

Bernard Paul Sypniewski

O.B.O., Inc.

syp@jerseycape.com

## ABSTRACT

This paper discusses the relation between linguistic types and the valence of operators on the genotype level of Applicative Universal Grammar (AUG) as set forth in Shaumyan (1987) and subsequent work. I use Professor Shaumyan's suggestion that **t** and **s** may be treated as zero-place operators Shaumyan (1987:196). There is a relationship between the valence of an operator and its genotype. The relation between valence and genotype briefly explains the difference between types operators, e.g., modifiers and predicates. The motivation for the preliminary work presented in this paper is the construction of a parser which correctly identifies superposition. Superposition is a theory of AUG which says that, in certain contexts, an expression may have a stratified type rather than a simple type.

BEST COPY AVAILABLE

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)

☒ This document has been reproduced as received from the person or organization originating it.

☐ Minor changes have been made to improve reproduction quality.

• Points of view or opinions stated in this document do not necessarily represent official OERI position or policy.

PERMISSION TO REPRODUCE AND DISSEMINATE THIS MATERIAL HAS BEEN GRANTED BY

Bernard Sypniewski

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)



FL 025764

**The motivation for this paper**

Superposition is a novel theory first proposed in Shaumyan (1987) and developed in Shaumyan and Sypniewski (1995) and Sypniewski (1996). The primary type of an expression $E$ is the genotype which describes the most common function performed by $E$. A secondary type is the genotype which describes a function performed by $E$ on other occasions. $E$ may have one or more secondary types. Briefly, the theory of superposition claims that in certain contexts, an expression may have a stratified type, i.e., a secondary type overlaid on its primary type. As an example, take the word *fat*. The primary type of *fat* is **t** (term) because *fat* most often names something, as in *The ham did not have much fat on it*. In some contexts, *fat* can have a secondary type, such as **Ott**, overlaid on its primary type: *The fat cat sat on the mat*. We say that *fat* in *fat cat* has been superposed and has a stratified type < **t : Ott** >.

This is obviously not the view of traditional linguistic theories which posit that a word has a state such as adjective or noun, but not a stratified state such as <noun : adjective>. AUG postulates that all languages operate on two levels: the *phenotype* level, specific to a particular language, and the *genotype* level, universal to all languages. By designing a parsing system properly, we can model and test AUG's two level approach to language. In general terms, a phenotype parser (which can be thought of as a sophisticated linguistic analyzer or LA) needs two functions:

1. to determine the genotypes of each word in a sentence passed to it; and
2. to arrange the genotyped sentence in such a way that the genotype parser may efficiently parse the sentence.

The genotype parser needs two functions:

1. to take the genotyped sentence from the phenotype parser and apply all operators to expressions smaller than a sentence, thereby reducing such expressions to linguistic units with single types; and
2. apply the predicate to each unit so formed until an s-type results.

I have come to the conclusion that traditional parser theory is inadequate for the task of construction a parser which properly evaluates superposition. Traditionally, the first stage[1] of a parse hands a sentence to an LA to break the sentence into tokens. Once a sentence is tokenized, the tokens are passed to the parser proper which assigns types, as determined by its formal grammar, to each token. A parser, in effect, looks at a word and asks "What is the type of this word?" The method that is used to answer this question is not important. What is important is that a word is presumed to have a type as part of its make-up.

I have questioned this method of type assignment. In theory, an LA may be designed to do more than it is traditionally required to do. An LA can be designed to compare the entire sentence to a sentence pattern (the *initial pattern*). If there are differences between the sentence pattern, by which I mean a pattern of genotypes, and the pattern of types resulting from the first pass through the sentence, which assigns the primary type to each word in the sentence, the LA resolves the differences by assuming that, in the contexts in which the differences occur[2], the word uses superposition to stratify its secondary type on its primary type. In the sentence *The fat cat sat on the mat*, the hypothetical LA assigns the primary types to each word and constructs pattern #1:

(1) Ott t t OtOts Opp Ott t

where **Ott** is a term modifier, **t** is term, **OtOts** is a two place predicate, and **Opp** is a predicate modifier, which can also be written **OOtOtsOtOts**. I will use **Opp** for convenience. The above type assignments assume that each word has a type and that no words are part of a multi-word expression which has a single type[3]. For instance, if *sat on* formed a single expression, *sat on* would have a single type **OtOts**. There would be no separate word with the type **Opp**. We would have:

(1a) Ott t t OtOts Ott t

The LA can compare (1) to the initial sentence pattern in (2):

(2) Ott Ott t OtOts Opp Ott t

The obvious difference is that (1) has three terms and (2) has three term modifiers, as can be seen in the following comparison in which the difference is marked in bold face:

Ott **t** t OtOts Opp Ott t

Ott **Ott** t OtOts Opp t

As we have seen, *fat*, the word in the bold faced context, has a secondary type of **Ott**. The hypothetical LA can resolve the difference between (1) and (2) by assigning *fat* a stratified type <**t : Ott**> that results from a superposition caused by the context. **Ott** and <**t : Ott**> are both acceptable and semiotically equivalent in this context. A primary type *x* and a secondary type *x* superposed on a primary type *y* are interchangeable because they both perform the same grammatical function, i.e., they both have the same grammatical meaning Shaumyan and Sypniewski (1995) We may say that, despite the additional information (primarily lexical) provided by the now-stratified primary type *y*, *x* and <*y* : *x*> are semiotically equivalent because they are interchangeable in the same context. The hypothetical LA need not compare (1) to any other sentence patterns.

Some types are highly contextually sensitive, e.g., conjunctions. A conjunction can join together expressions of any type. If a genotype is a description of the role or function of an expression, a conjunction, in theory, can have as many different genotypes as there are combinations of genotypes. Even if we were to limit the number of genotypes to the ten most basic types (**s**, **t**, **Oss**, **Ott**, **Ots**, **OtOts**, **OtOtOts**, **OOtsOts**, **OOtOtsOtOts**, and **OOtOtOtsOtOtOts**), a conjunction can have 2^10 genotypes. This is obviously unacceptable since the number of choices can only degrade the performance of a parser. We know that the "parser" in the human linguistic system is, essentially, instantaneous. The speed of the human parsing system is given short shrift by most linguists; in fact, it is a major constraint on any linguistic system.

Shaumyan suggests that **t** and **s** can be thought of as zero-place operators. Shaumyan (1987:196) This means that **t** and **s** can also be written as **O**. Another way might be **0→0**. The latter notation form highlights the *valence* of the expression. All operators in AUG and combinatory logic, its underlying formalism, are one-place operators. Some operators take more than one step to complete their operation. The valence of an operator is the number of steps that an operator takes to complete its operation. Valence notation is in the form **n→m**, where **n** is the valence of the operand and **m** is the valence of the resultant. A term can be noted as **0→0** because it takes no operand and, therefore, cannot produce a resultant. A term modifier, **Ott**, also has an

valence notation of **0→0** because its operand is a zero-place operator (a term) and the resultant is also a zero-place operator (a term). To distinguish between a term and a sentence, both of which are zero-place operators, let us say that the valence of **s** is -1 by which we understand that **s** is a zero-place operator. We simply use the notation -1 as a visual distinction between **s** and **t**. So, a sentence modifier **Oss** has an valence notation of **-1→ -1**.

Predicate modifiers, while a bit more complex, also have an valence notation like term modifiers. A predicate modifier which modifies a one-place predicate **OOtsOts** has an valence notation of **1→1**. A predicate modifier which modifies a two-place predicate **OOtOtsOtOts** has an valence notation of **2→2**. A predicate modifier which modifies a three-place predicate **OOtOtOtsOtOtOts** has an valence notation of **3→3**.

Predicates reduce their operands to a different type. A one-place predicate **Ots** has an valence notation of **0 → -1** because a one-place predicate takes a term (zero-place operator) and results in an s-type (sentence). A two place predicate **OtOts** is **0→1** because it is a two-place operator which takes a term (zero-place operator) and results in a one-place operator **Ots**. A three-place predicate **OtOtOts** has an valence notation of **0→2** because it takes a term (zero-place operator) and results in a two-place operator **OtOts.**

One difference between a predicate and a modifier is that, for a modifier, the valence of the operand, $A_o$, equals the valence of the resultant, $A_r$. For a predicate, $A_o \neq A_r$. The difference in valence relations between a modifier and a predicate may be due to the fact that modification changes only the semantic (lexical) meaning of the type being modified while predication changes both the grammatical and semantic meanings of the expression. In the sentence *John bought a new watch*, *new* modifies *watch*. The modification tells us something about the term *watch* but after the operation, *new watch* is still a term. *Bought*, a two-place operator, becomes *bought a new watch* after being applied to the secondary term (*a new watch*). The action of buying a new watch is certainly different from the action of buying something unspecified. The application not only results in a new semantic meaning but results in a one-place, rather than a two-place,

predicate. After *bought a new watch* is applied to the primary term *John*, a sentence, which is a zero-place operator, results.

We can posit two theories from these observations:

**The theory of modification**: Modification is an operation which changes the lexical meaning of an expression but not the grammatical meaning of an expression.

**The theory of predication:** Predication is an operation which effects both the lexical and grammatical meanings of an expression.

The primary and secondary types of a word differ from each other. Many terms have a secondary type of **Ott**, e.g., *chicken* in *Why did the chicken cross the road?* and *chicken* as in *chicken sandwich*. Others have a secondary type which is a predicate, e.g., *fish* and *to fish*. What is not always obvious is which is the primary and which is the secondary type. This presents an obvious difficulty for any computerized parser. Current parser theory does not rely on word behavior but, rather, on the notion that types are static, in order to assign types to words. AUG postulates that there are no invariant types or that invariant types are rare. It is word behavior which determines the type of a word. Indeed, a type is nothing more than a convenient description of word behavior.

In order to construct a parser which properly assigns genotypes to words, information about the primary and secondary types of words is necessary. However, since AUG is a new theory without the number of workers that other theories enjoy, primary and secondary type data is not readily available. An important stage of an AUG parser project is the gathering of preliminary primary and secondary type information. In my preparation for gathering such material, I have confronted the theoretically important issue of whether it is even possible to generalize about types. In other words, can statements such as *terms generally have secondary types of **Ott*** be meaningfully made? It is at least possible that the notion of primary and secondary types must be a context sensitive notion. For example, in a discussion about tropical fish, *fish* may have a primary type of term. In a discussion about the fishing industry, *fish* may have a primary type of predicate. Can we

even hazard what the *overall* primary type of the word *fish* is, given the scope and amount of human discourse? Yes, but the answer to the question whether we can speak generally about the relations between types has interesting implications for the construction of a parser. We can create a simple test to demonstrate that, in principle, every word has term as either its primary or secondary type. If we form the question skeleton

What does *w* mean?

We can substitute any word in any language for *w* and have a coherent sentence. *W* performs the function of term. However, this test does not answer the question "Does *w* have **t** as its primary type?" We could answer the question in one of two ways.

1. We could take the position that any word which even arguably names something has the primary type of term.
2. We can take a statistical approach and say that *w* has the primary type of term if *w* is most frequently a term in common usage.

Position (1) leads us to ask "What does it mean to name something?", which, I contend, is tantamount to asking "What is a term?". I don't think that it gets us very far. Position (2) is also fraught with difficulties. It is obviously relativistic and the question now arises "Relative to what?" What is "common usage"? At present, this question is theoretical.

Assume than a substantial amount of primary and secondary type data has been gathered about English words. Can we then project the findings of an analysis of this data onto another language? Let us assume that the data that is collected is limited to specific word forms. We gather data about the primary and secondary types of the word *fish* separately from the words *fishing* or *fishes*, for example. We are confronted by a similar collection attempt in an inflected language such as Latin. It is apparent even for someone with a slight familiarity with Latin that whatever conclusions may be drawn from the English data about the word *fish* would not be applicable to the Latin equivalent since the noun and verbs forms are substantially different. Assuming that this observation, that inflected and analytic languages may show substantial differences in the relations between primary and secondary types, what conclusions can be drawn? Is it possible that the notion

of word types itself must be reconsidered? After all, word types developed in the context of inflected languages. The Latin word *nomen* can be typed as a term (and as a primary term) with much more assurance than can the English word *name*. Even in Latin, it is not always apparent what the primary type of a particular word is. Many words with potentially different types (dative and ablative plurals, for instance) often have the same form. The distinction between such forms may be more traditional than justified.

Genotype grammar, especially its theory of superposition, has radically altered our perceptions of the inner workings of language. Language has become a behavioral mechanism, i.e., a dynamic mechanism. New questions need to be asked and new methods of answering questions need to be found. The tools and techniques of traditional grammatical systems may not be adequate to the task.

---

[1] I describe a parse within the context of a computerized parse. My goal is to create a computer program which correctly identifies superposition.

[2] By *context*, I mean the word or unit with the type different from that predicted by the initial pattern. For example, if the initial pattern predicts that the word *fat* is **t** but *fat* with type **t** does not make linguistic sense, the context is the word *fat*. The type for the word *fat* in this context needs to be resolved.

[3] I have elsewhere referred to such constructions as *units*.